Logische Methoden des Software Engineerings
Vertiefungsmodul 1

Inhabitation in $\lambda^{\rightarrow}$

Jakob Rehof & Andrej Dudenhefner
LS XIV – Software Engineering

TU Dortmund
WS 2018/19

WS 2018/19

Inhabitation

- Whenever $\vdash M : \tau$, we say that M is an *inhabitant* in (or, of) τ.
- The *inhabitation problem* is concerned with the existence of inhabitants in a given type: Given a type, is there a term having the type?
- Notice that this problem is dual to the typability problem: Given a term, does it have a type?

Curry-Howard isomorphism

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}(\text{var})$$

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}(\to\text{I})$$

$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \sigma}(\to\text{E})$$

Curry-Howard isomorphism

$$\frac{}{\Gamma, \tau \vdash \tau}\text{(hyp)}$$

$$\frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \to \sigma}\text{(DT)}$$

$$\frac{\Gamma \vdash \tau \to \sigma \quad \Gamma \vdash \tau}{\Gamma \vdash \sigma}\text{(MP)}$$

Inhabitation Problem

The *inhabitation problem* is the following decision problem:

Definition 1 (Inhabitation problem)

- Given Γ and τ, does there exist M such that $\Gamma \vdash M : \tau$?

Inhabitation Problem

The *inhabitation problem* is the following decision problem:

Definition 1 (Inhabitation problem)

- Given Γ and τ, does there exist M such that $\Gamma \vdash M : \tau$?

Remark 1

Notice that the inhabitation problem is equivalent to the following restricted problem:

- *Given* Γ *and* τ*, does there exist a* normal form N *such that* $\Gamma \vdash N : \tau$*?*

The *inhabitation problem* is the following decision problem:

Definition 1 (Inhabitation problem)

- Given Γ and τ, does there exist M such that $\Gamma \vdash M : \tau$?

Remark 1

Notice that the inhabitation problem is equivalent to the following restricted problem:

- *Given Γ and τ, does there exist a* normal form *N such that $\Gamma \vdash N : \tau$?*

Remark 2

Notice that inhabitation is equivalent to provability *in implicational intuitionistic propositional logic.*

Inhabitation

$$\vdash \; ? : (a \to c) \to (b \to a \to c) \to a \to b \to c$$

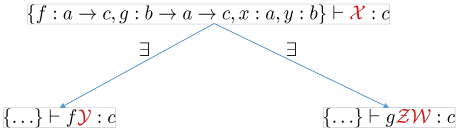
Inhabitation

$$\vdash \; ? : (a \to c) \to (b \to a \to c) \to a \to b \to c$$

$$\{f : a \to c, g : b \to a \to c, x : a, y : b\} \vdash \mathcal{X} : c$$

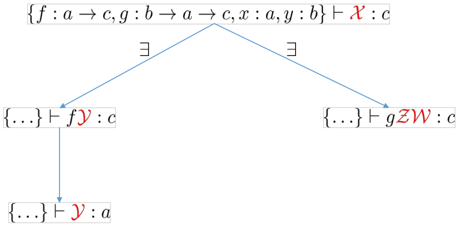
Inhabitation

$$\vdash \, ? : (a \to c) \to (b \to a \to c) \to a \to b \to c$$



Inhabitation

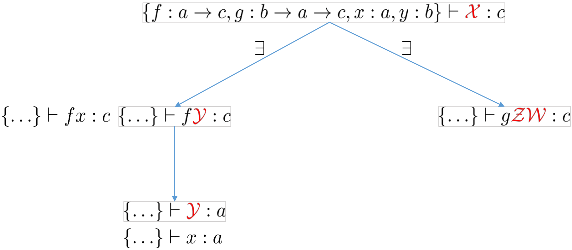
$$\vdash \, ? : (a \to c) \to (b \to a \to c) \to a \to b \to c$$



Inhabitation

$$\vdash {\color{red}?} : (a \to c) \to (b \to a \to c) \to a \to b \to c$$

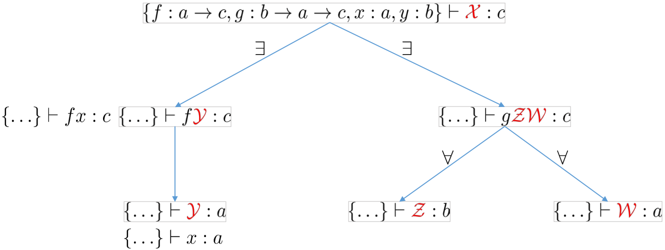
$\vdash \lambda f.\lambda g.\lambda x.\lambda y.fx : \sigma$



Inhabitation

$$\vdash {\color{red}?} : (a \to c) \to (b \to a \to c) \to a \to b \to c$$

$\vdash \lambda f.\lambda g.\lambda x.\lambda y.fx : \sigma$

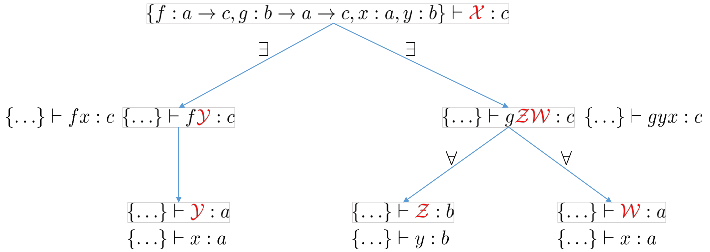


Inhabitation

$$\vdash ? : (a \to c) \to (b \to a \to c) \to a \to b \to c$$

$\vdash \lambda f.\lambda g.\lambda x.\lambda y.fx : \sigma$

$\vdash \lambda f.\lambda g.\lambda x.\lambda y.gyx : \sigma$



Alternating Turing machines (ATM)

An *alternating Turing machine* is a tuple $\mathcal{M} = (\Sigma, Q, q_0, q_a, q_r, \Delta)$. The set of states $Q = Q_\exists \uplus Q_\forall$ is partitioned into a set $Q_\exists$ of existential states and a set $Q_\forall$ of universal states. There is an initial state $q_0 \in Q$, an accepting state $q_a \in Q_\forall$, and a rejecting state $q_r \in Q_\exists$. We take $\Sigma = \{0, 1, \textvisiblespace\}$, where $\textvisiblespace$ is the blank symbol (used to initialize the tape but not written by the machine).

The transition relation Δ satisfies

$$\Delta \subseteq \Sigma \times Q \times \Sigma \times Q \times \{\mathrm{L}, \mathrm{R}\},$$

where $h \in \{\mathrm{L}, \mathrm{R}\}$ are the moves of the machine head (left and right). For $b \in \Sigma$ and $q \in Q$, we write $\Delta(b, q) = \{(c, p, h) \mid (b, q, c, p, h) \in \Delta\}$. We assume $\Delta(b, q_a) = \Delta(b, q_r) = \emptyset$, for all $b \in \Sigma$, and $\Delta(b, q) \neq \emptyset$ for $q \in Q \setminus \{q_a, q_r\}$.

Alternating Turing machines (ATM)

A *configuration* $\mathcal{C}$ of $\mathcal{M}$ is a word wqw' with $q \in Q$ and $w, w' \in \Sigma^*$. The *successor* relation $\mathcal{C} \Rightarrow \mathcal{C}'$ on configurations is defined as usual, according to Δ. We classify a configuration wqw' as *existential*, *universal*, *accepting* etc., according to q.

The notion of *eventually accepting* configuration is defined by induction (i.e., the set of all eventually accepting configurations is the smallest set satisfying the following closure conditions):

- An accepting configuration is eventually accepting.
- If $\mathcal{C}$ is existential and some successor of $\mathcal{C}$ is eventually accepting then so is $\mathcal{C}$.
- If $\mathcal{C}$ is universal and all successors of $\mathcal{C}$ are eventually accepting then so is $\mathcal{C}$.

Alternating Turing machines (ATM)

We use the notation for instruction sequences starting from existential states

- CHOOSE $x \in A$

and instruction sequences starting from universal states

- FORALL $(i = 1 \ldots k)\, S_i$

A command of the form CHOOSE $x \in A$ branches from an existential state to successor states in which x gets assigned distinct elements of A. A command of the form FORALL $(i = 1 \ldots k)\, S_i$ branches from a universal state to successor states from which each instruction sequence S_i is executed.

Alternating complexity

Some alternating complexity classes:

- $\mathrm{APTIME} := \bigcup_{k>0} \mathrm{ATIME}(n^k)$
- $\mathrm{APSPACE} := \bigcup_{k>0} \mathrm{ASPACE}(n^k)$
- $\mathrm{AEXPTIME} := \bigcup_{k>0} \mathrm{ATIME}(k^n)$

Theorem 2 (Chandra, Kozen, Stockmeyer 1981)

- $\mathrm{APTIME} = \mathrm{PSPACE}$
- $\mathrm{APSPACE} = \mathrm{EXPTIME}$
- $\mathrm{AEXPTIME} = \mathrm{EXPSPACE}$

Inhabitation in $\lambda^{\rightarrow}$ is PSPACE-complete

We will give a detailed proof of Statman's Theorem: inhabitation in $\lambda^{\rightarrow}$ is PSPACE-complete. This result was first proven in [Sta79] (using, among other things, results of Ladner [Lad77]).

Our proof follows [Urz97] (see also [SU06]) where a syntactic approach was used, and where alternation is used to simplify the proof.

Inhabitation in $\lambda^{\to}$: upper bound

Notice that every type τ of $\lambda^{\to}$ can be written on the form $\tau \equiv \tau_1 \to \cdots \tau_n \to a$, $n \geq 0$, where a is an atom (either a type variable or a type constant).

Notice that every application context can be written on the form $xP_1 \cdots P_n$ for some maximal $n \geq 0$.

An explicitly typed λ-term M is in η*-long normal form* if it is a β-normal form and every maximal application in M has the form $x^{\tau_1 \to \cdots \to \tau_n \to a} P_1^{\tau_1} \cdots P_n^{\tau_n}$. In other words, in such terms applications are fully applied according to the type of the operator.

Notice that every typed β-normal form of type τ can be converted into η*-long normal form*: any subterm occurrence of a maximal application $Q^{\sigma \to \rho}$ can be converted into $\lambda x : \sigma.Qx$ where $x \notin \mathsf{FV}(Q)$.

Set $\Gamma \boxplus (x : \tau) = \Gamma$, if there exists $y \in \mathsf{Dm}(\Gamma)$ with $\Gamma(y) = \tau$, and otherwise $\Gamma \boxplus (x : \tau) = \Gamma \cup \{(x : \tau)\}$.

Inhabitation in $\lambda^{\rightarrow}$: upper bound

Algorithm INH($\lambda^{\rightarrow}$)

Input : Γ, τ

loop :
IF $(\tau \equiv a)$
THEN
    CHOOSE $(x : \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow a) \in \Gamma$;
    IF $(n = 0)$ THEN ACCEPT;
    ELSE
        FORALL $(i = 1 \ldots n)$
            $\tau := \sigma_i$;
            GOTO *loop*;
ELSE IF $(\tau \equiv \sigma \rightarrow \rho)$
THEN
    $\Gamma := \Gamma \boxplus (y : \sigma)$ where y is fresh;
    $\tau := \rho$;
    GOTO *loop*;

Proposition 1

Inhabitation in $\lambda^{\rightarrow}$ is in PSPACE.

Proof.

By algorithm INH($\lambda^{\rightarrow}$). Clearly, the algorithm performs exhaustive search for η-long normal form inhabitants. The algorithm decides inhabitation in $\lambda^{\rightarrow}$ in polynomial space. For consider configurations (Γ, τ) arising during an entire run of the algorithm on input (Γ_0, τ_0). Notice that Γ and τ always only contain types that are subtrees of types present in the previous values of Γ and τ (line 7 and line 11). Since a tree of size m has m distinct subtrees, the set of distinct configurations (Γ, τ) can be bounded by n^2, where n is the size of the input. Hence, the algorithm shows that the problem is in APTIME, which is PSPACE by Theorem 2. □

Inhabitation in $\lambda^{\to}$: lower bound

Reduction from provability of quantified boolean fomulae ϕ, χ, ψ:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall p.\phi \mid \exists p.\phi$$

We can assume w.l.o.g. that negation is only applied to propositional variables p in ϕ, that all bound variables are distinct and that no variable occurs both free and bound.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.

Inhabitation in $\lambda^{\to}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \neg p$, then $\Gamma_\phi = \emptyset$.

Inhabitation in $\lambda^{\to}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \neg p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \chi \wedge \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x_\phi : \alpha_\chi \to \alpha_\psi \to \alpha_{\chi \wedge \psi}\}$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \neg p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \chi \wedge \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x_\phi : \alpha_\chi \rightarrow \alpha_\psi \rightarrow \alpha_{\chi \wedge \psi}\}$.
- If $\phi \equiv \chi \vee \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x^l_\phi : \alpha_\chi \rightarrow \alpha_{\chi \vee \psi}, x^r_\phi : \alpha_\psi \rightarrow \alpha_{\chi \vee \psi}\}$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \neg p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \chi \wedge \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x_\phi : \alpha_\chi \to \alpha_\psi \to \alpha_{\chi\wedge\psi}\}$.
- If $\phi \equiv \chi \vee \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x^l_\phi : \alpha_\chi \to \alpha_{\chi\vee\psi}, x^r_\phi : \alpha_\psi \to \alpha_{\chi\vee\psi}\}$.
- If $\phi \equiv \forall p.\psi$, then $\Gamma_\phi = \Gamma_\psi \cup \{x_\phi : (\alpha_p \to \alpha_\psi) \to (\alpha_{\neg p} \to \alpha_\psi) \to \alpha_{\forall p.\psi}\}$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Given formula ϕ, construct type environment Γ_ϕ by induction on ϕ:

- For each propositional variable p in ϕ, let α_p and $\alpha_{\neg p}$ be fresh type variables. For each subformula ψ, let α_ψ be fresh type variables.
- If $\phi \equiv p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \neg p$, then $\Gamma_\phi = \emptyset$.
- If $\phi \equiv \chi \wedge \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x_\phi : \alpha_\chi \rightarrow \alpha_\psi \rightarrow \alpha_{\chi \wedge \psi}\}$.
- If $\phi \equiv \chi \vee \psi$, then $\Gamma_\phi = \Gamma_\chi \cup \Gamma_\psi \cup \{x^l_\phi : \alpha_\chi \rightarrow \alpha_{\chi \vee \psi}, x^r_\phi : \alpha_\psi \rightarrow \alpha_{\chi \vee \psi}\}$.
- If $\phi \equiv \forall p.\psi$, then $\Gamma_\phi = \Gamma_\psi \cup \{x_\phi : (\alpha_p \rightarrow \alpha_\psi) \rightarrow (\alpha_{\neg p} \rightarrow \alpha_\psi) \rightarrow \alpha_{\forall p.\psi}\}$.
- If $\phi \equiv \exists p.\psi$, then
  $\Gamma_\phi = \Gamma_\psi \cup \{x^0_\phi : (\alpha_p \rightarrow \alpha_\psi) \rightarrow \alpha_{\exists p.\psi}, x^1_\phi : (\alpha_{\neg p} \rightarrow \alpha_\psi) \rightarrow \alpha_{\exists p.\psi}\}$.

Inhabitation in $\lambda^{\to}$: lower bound

A valuation v is a map from propositional variables to truth values in $\{0, 1\}$.

Inhabitation in $\lambda^{\to}$: lower bound

A valuation v is a map from propositional variables to truth values in $\{0, 1\}$.

For a formula ϕ and a valuation v, let Γ_ϕ^v be the extension of Γ_ϕ:

$$\Gamma_\phi^v = \Gamma_\phi \cup \bigcup_{p \in \mathsf{Dm}(v)} \{x_p : \langle\alpha\rangle_v^p\}$$

where $\langle\alpha\rangle_v^p = \alpha_p$ if $v(p) = 1$ and $\langle\alpha\rangle_v^p = \alpha_{\neg p}$ if $v(p) = 0$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

A valuation v is a map from propositional variables to truth values in $\{0, 1\}$.

For a formula ϕ and a valuation v, let Γ_ϕ^v be the extension of Γ_ϕ:

$$\Gamma_\phi^v = \Gamma_\phi \cup \bigcup_{p \in \mathsf{Dm}(v)} \{x_p : \langle\alpha\rangle_v^p\}$$

where $\langle\alpha\rangle_v^p = \alpha_p$ if $v(p) = 1$ and $\langle\alpha\rangle_v^p = \alpha_{\neg p}$ if $v(p) = 0$.

A valuation of a formula ϕ is a valuation defined on the free variables of ϕ.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

A valuation v is a map from propositional variables to truth values in $\{0, 1\}$.

For a formula ϕ and a valuation v, let Γ_ϕ^v be the extension of Γ_ϕ:

$$\Gamma_\phi^v = \Gamma_\phi \cup \bigcup_{p \in \mathsf{Dm}(v)} \{x_p : \langle\alpha\rangle_v^p\}$$

where $\langle\alpha\rangle_v^p = \alpha_p$ if $v(p) = 1$ and $\langle\alpha\rangle_v^p = \alpha_{\neg p}$ if $v(p) = 0$.

A valuation of a formula ϕ is a valuation defined on the free variables of ϕ.

We write $v \oplus [p := b]$ for the extension of v mapping p to $b \in \{0, 1\}$.

Inhabitation in $\lambda^{\to}$: lower bound

A valuation v is a map from propositional variables to truth values in $\{0, 1\}$.

For a formula ϕ and a valuation v, let Γ_ϕ^v be the extension of Γ_ϕ:

$$\Gamma_\phi^v = \Gamma_\phi \cup \bigcup_{p \in \mathsf{Dm}(v)} \{x_p : \langle\alpha\rangle_v^p\}$$

where $\langle\alpha\rangle_v^p = \alpha_p$ if $v(p) = 1$ and $\langle\alpha\rangle_v^p = \alpha_{\neg p}$ if $v(p) = 0$.

A valuation of a formula ϕ is a valuation defined on the free variables of ϕ.

We write $v \oplus [p := b]$ for the extension of v mapping p to $b \in \{0, 1\}$.

We write $\Gamma \not\vdash \tau$ as abbreviation for $\neg\exists M.\ \Gamma \vdash M : \tau$.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$[\![p]\!]v \quad = \quad v(p)$$

Inhabitation in $\lambda^{\to}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$\begin{array}{lcl} [\![p]\!]v & = & v(p) \\ [\![\neg p]\!]v & = & 0, \text{if } v(p) = 1, \text{else } 1 \end{array}$$

Inhabitation in $\lambda^{\to}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$
\begin{array}{lcl}
[\![p]\!]v & = & v(p) \\
[\![\neg p]\!]v & = & 0, \text{if } v(p) = 1, \text{else } 1 \\
[\![\psi \wedge \chi]\!]v & = & \min\{[\![\psi]\!]v, [\![\chi]\!]v\}
\end{array}
$$

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$
\begin{array}{lcl}
[\![p]\!]v & = & v(p) \\
[\![\neg p]\!]v & = & 0, \text{if } v(p) = 1, \text{else } 1 \\
[\![\psi \wedge \chi]\!]v & = & \min\{[\![\psi]\!]v, [\![\chi]\!]v\} \\
[\![\psi \vee \chi]\!]v & = & \max\{[\![\psi]\!]v, [\![\chi]\!]v
\end{array}
$$

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$
\begin{array}{lcl}
[\![p]\!]v & = & v(p) \\
[\![\neg p]\!]v & = & 0, \text{if } v(p) = 1, \text{else } 1 \\
[\![\psi \wedge \chi]\!]v & = & \min\{[\![\psi]\!]v, [\![\chi]\!]v\} \\
[\![\psi \vee \chi]\!]v & = & \max\{[\![\psi]\!]v, [\![\chi]\!]v\} \\
[\![\forall p.\psi]\!]v & = & \min\{[\![\psi]\!](v \oplus [p := 1]), [\![\psi]\!](v \oplus [p := 0])
\end{array}
$$

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Assume w.l.o.g. that formulae ϕ have negation signs only applied to propositional variables.

We let $[\![\phi]\!]v$ denote the truth value of ϕ under valuation v, defined by induction on ϕ:

$$
\begin{array}{lcl}
[\![p]\!]v & = & v(p) \\
[\![\neg p]\!]v & = & 0, \text{if } v(p) = 1, \text{else } 1 \\
[\![\psi \wedge \chi]\!]v & = & \min\{[\![\psi]\!]v, [\![\chi]\!]v\} \\
[\![\psi \vee \chi]\!]v & = & \max\{[\![\psi]\!]v, [\![\chi]\!]v\} \\
[\![\forall p.\psi]\!]v & = & \min\{[\![\psi]\!](v \oplus [p := 1]), [\![\psi]\!](v \oplus [p := 0])\} \\
[\![\exists p.\psi]\!]v & = & \max\{[\![\psi]\!](v \oplus [p := 1]), [\![\psi]\!](v \oplus [p := 0])\}
\end{array}
$$

Inhabitation in $\lambda^{\to}$: lower bound

Lemma 3

For every formula ϕ and every valuation v of ϕ, one has

$$[\![\phi]\!]v = 1 \;\Leftrightarrow\; \exists M.\, \Gamma_\phi^v \vdash M : \alpha_\phi$$

Proof

By induction on ϕ.

Case $\phi \equiv p$. If $[\![p]\!]v = 1$, i.e., $v(p) = 1$, then $\Gamma_\phi^v = \{x_p^v : \alpha_p\}$, so $\Gamma_\phi^v \vdash x_p^v : \alpha_p$. If $\Gamma_\phi^v \vdash M : \alpha_p$, then, by construction of Γ_ϕ^v, it must be the case that $\Gamma_\phi^v = \{x_p^v : \alpha_p\}$, so that $v(p) = 1$.

Case $\phi \equiv \neg p$. Similar to previous case.

Proof (continued)

Case $\phi \equiv \chi \wedge \psi$

If $[\![\phi]\!]v = 1$, then $[\![\chi]\!]v = [\![\psi]\!]v = 1$. By induction hypothesis, $\Gamma^v_\chi \vdash M : \alpha_\chi$ and $\Gamma^v_\psi \vdash N : \alpha_\psi$, for some M and N. It follows that $\Gamma^v_{\chi\wedge\psi} \vdash x_{\chi\wedge\psi} M N : \alpha_{\chi\wedge\psi}$.

If $[\![\phi]\!]v = 0$, then $[\![\chi]\!]v = 0$ or $[\![\psi]\!]v = 0$. If $[\![\chi]\!]v = 0$, then by induction hypothesis, $\Gamma^v_\chi \not\vdash \alpha_\chi$, hence by construction of Γ^v_ϕ, we must have $\Gamma^v_\phi \not\vdash \alpha_\chi$. It follows that $\Gamma^v_\phi \not\vdash \alpha_{\chi\wedge\psi}$. The case where $[\![\psi]\!]v = 0$ is analogous.

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Proof (continued)

Case $\phi \equiv \forall p.\psi$

If $[\![\phi]\!]v = 1$, then $[\![\psi]\!]v_0 = [\![\psi]\!]v_1 = 1$, where $v_0 = v \oplus [p := 0]$ and $v_1 = v \oplus [p := 1]$. By induction hypothesis, we have $\Gamma_\psi^{v_0} \vdash M : \alpha_\psi$ and $\Gamma_\psi^{v_1} \vdash N : \alpha_\psi$, for some M and N, which (by definitions) can also be written as $\Gamma_\phi^v \cup \{x_p : \alpha_{\neg p}\} \vdash M : \alpha_\psi$ and $\Gamma_\phi^v \cup \{x_p : \alpha_p\} \vdash N : \alpha_\psi$. Hence, $\Gamma_\phi^v \vdash \lambda x_p : \alpha_{\neg p}.M : \alpha_{\neg p} \rightarrow \alpha_\psi$ and $\Gamma_\phi^v \vdash \lambda x_p : \alpha_p.N : \alpha_p \rightarrow \alpha_\psi$. It follows that we have

$$\Gamma_\phi^v \vdash x_\phi(\lambda x_p : \alpha_p.N)(\lambda x_p : \alpha_{\neg p}.M) : \alpha_\phi$$

Inhabitation in $\lambda^{\rightarrow}$: lower bound

Proof (continued)

Case $\phi \equiv \forall p.\psi$

If $[\![\phi]\!]v = 0$, then either we have $[\![\psi]\!](v \oplus [p := 0]) = 0$ or $[\![\psi]\!](v \oplus [p := 1]) = 0$. Suppose that the former is the case. Then, by induction hypothesis, we have $\Gamma_{\psi}^{v_0} \not\vdash \alpha_{\psi}$, where $v_0 = v \oplus [p := 0]$. Hence, by definitions, we have $\Gamma_{\psi} \cup \{x_p : \alpha_{\neg p}\} \not\vdash \alpha_{\psi}$. By construction of Γ_{ϕ}^{v}, it follows that we have $\Gamma_{\phi}^{v} \not\vdash \alpha_{\phi}$. The case where $[\![\psi]\!](v \oplus [p := 1]) = 0$ is analogous.

Proof (continued)

Remaining cases are left as an exercise :)

Proposition 2

Inhabitation in $\lambda^{\to}$ is PSPACE*-hard.*

Proof.

In order to decide provability of QBF formula ϕ, it suffices to ask whether $\Gamma_\phi \vdash ? : \alpha_\phi$, by Lemma 3. Since the construction of Γ_ϕ can be carried out in logarithmic space, the proposition follows from PSPACE-hardness of QBF. □

Theorem 4 (Statman 1979)

Inhabitation in $\lambda^{\to}$ is PSPACE*-complete.*

Proof.

By Proposition 1 and Proposition 2. $\square$

R.E. Ladner.
The Computational Complexity of Provability in Systems of Modal Propositional Logic.
SIAM J. Comput., 6(3):467 – 480, 1977.

Richard Statman.
Intuitionistic Propositional Logic is Polynomial-space Complete.
Theoretical Computer Science, 9:67–72, 1979.

M.H. Sørensen and P. Urzyczyn.
Lectures on the Curry-Howard Isomorphism, volume 149 of *Studies in Logic and the Foundations of Mathematics*.
Elsevier, 2006.

P. Urzyczyn.
Inhabitation in Typed Lambda-Calculi (A Syntactic Approach).
In *TLCA'97, Typed Lambda Calculi and Applications, Proceedings*, volume 1210 of *LNCS*, pages 373–389. Springer, 1997.